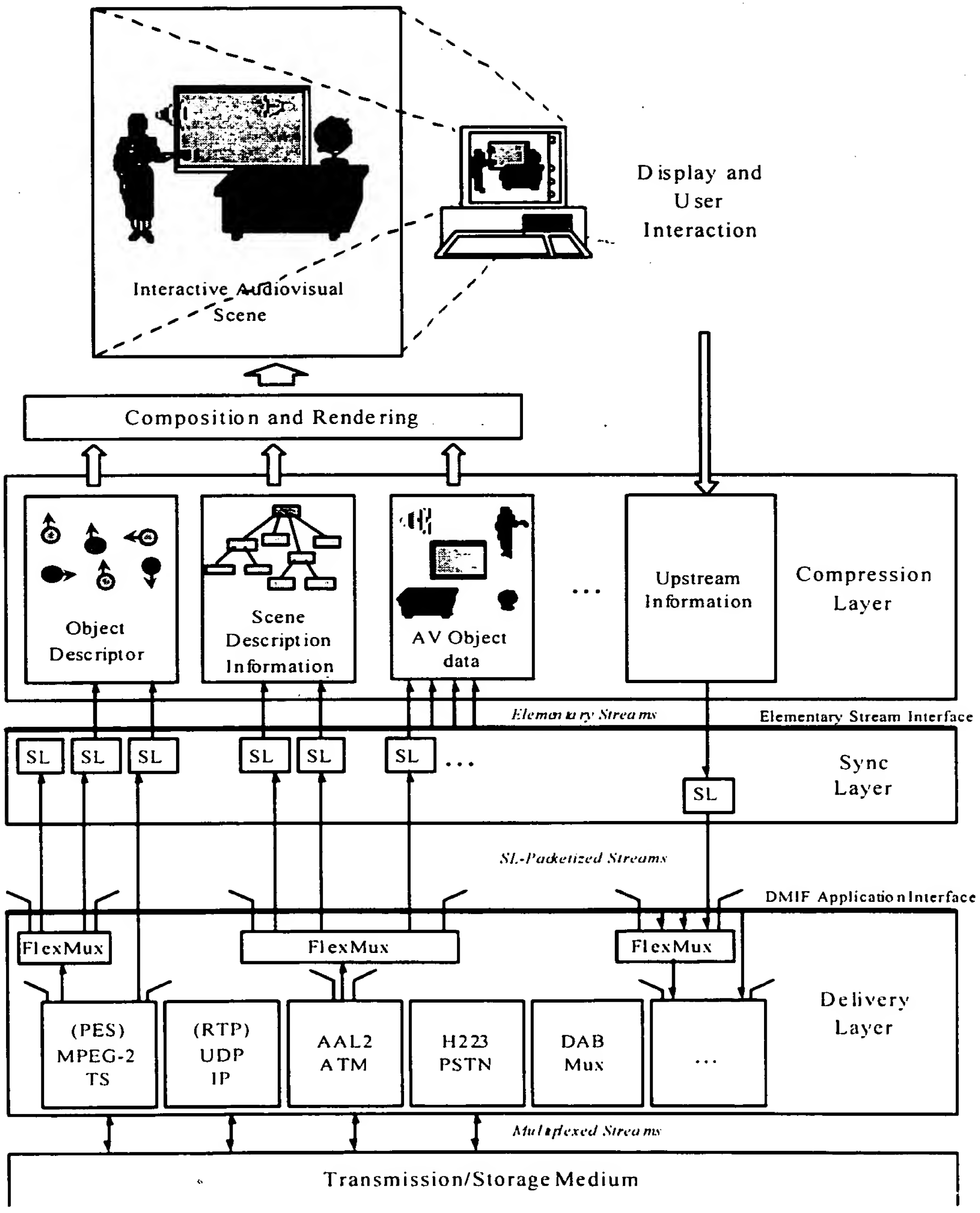


Interactive Audiovisual Scene
Display and User Interaction
Composition and Rendering
Object Descriptor
Scene Description Information
AV Object data
...
Upstream Information
Compression Layer
Elementary Streams
Elementary Stream Interface
SL
SL
SL
SL
SL
SL
...
SL
Sync Layer
SL-Packetized Streams
DMIF Application Interface
FlexMux
FlexMux
FlexMux
(PES) MPEG-2 TS
(RTP) UDP IP
AAL2 ATM
H223 PSTN
DAB Mux
...
Delivery Layer
Multiplexed Streams
Transmission/Storage Medium

FIG. 1

6662T0"60S0h260

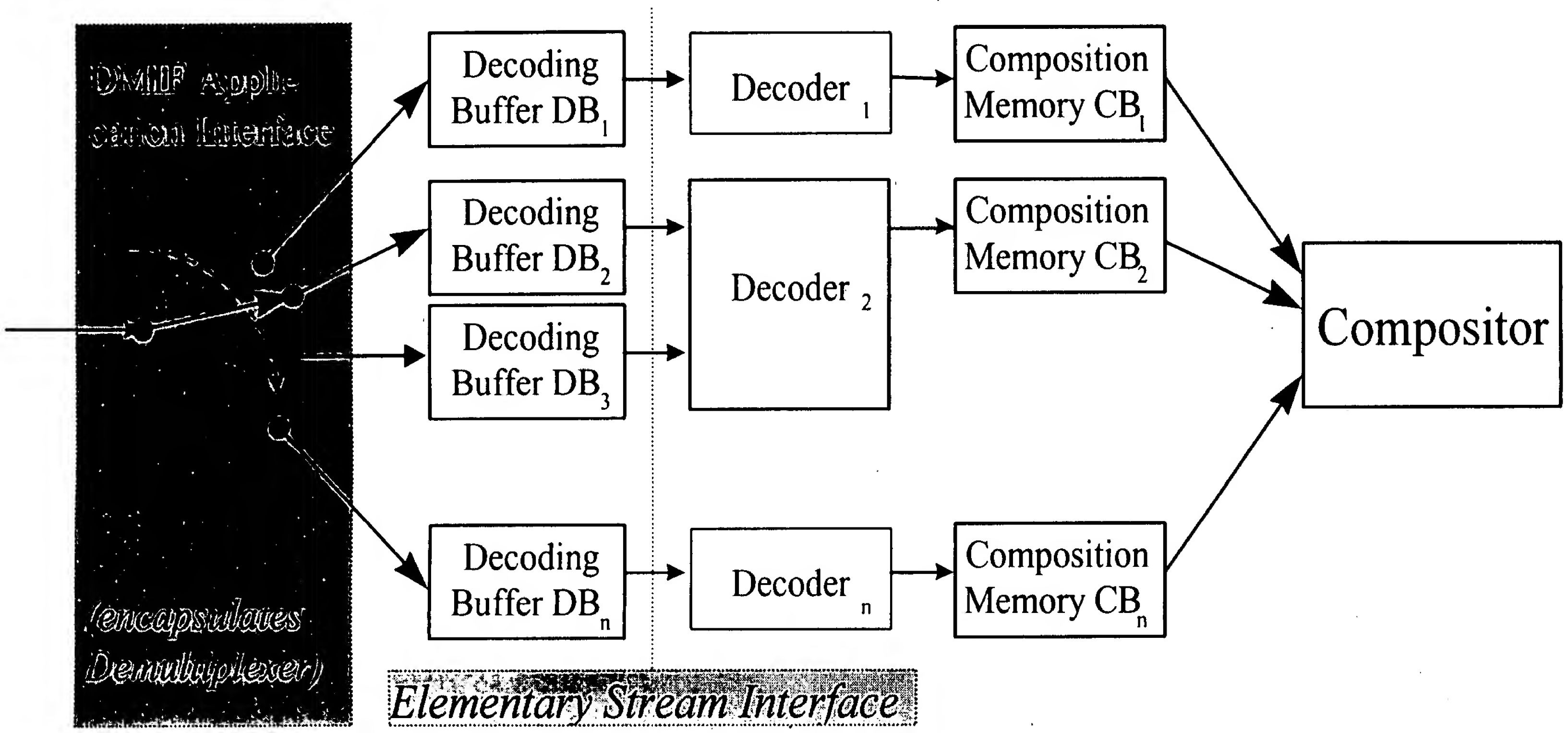


FIG. 2

66621 0 " 60504260

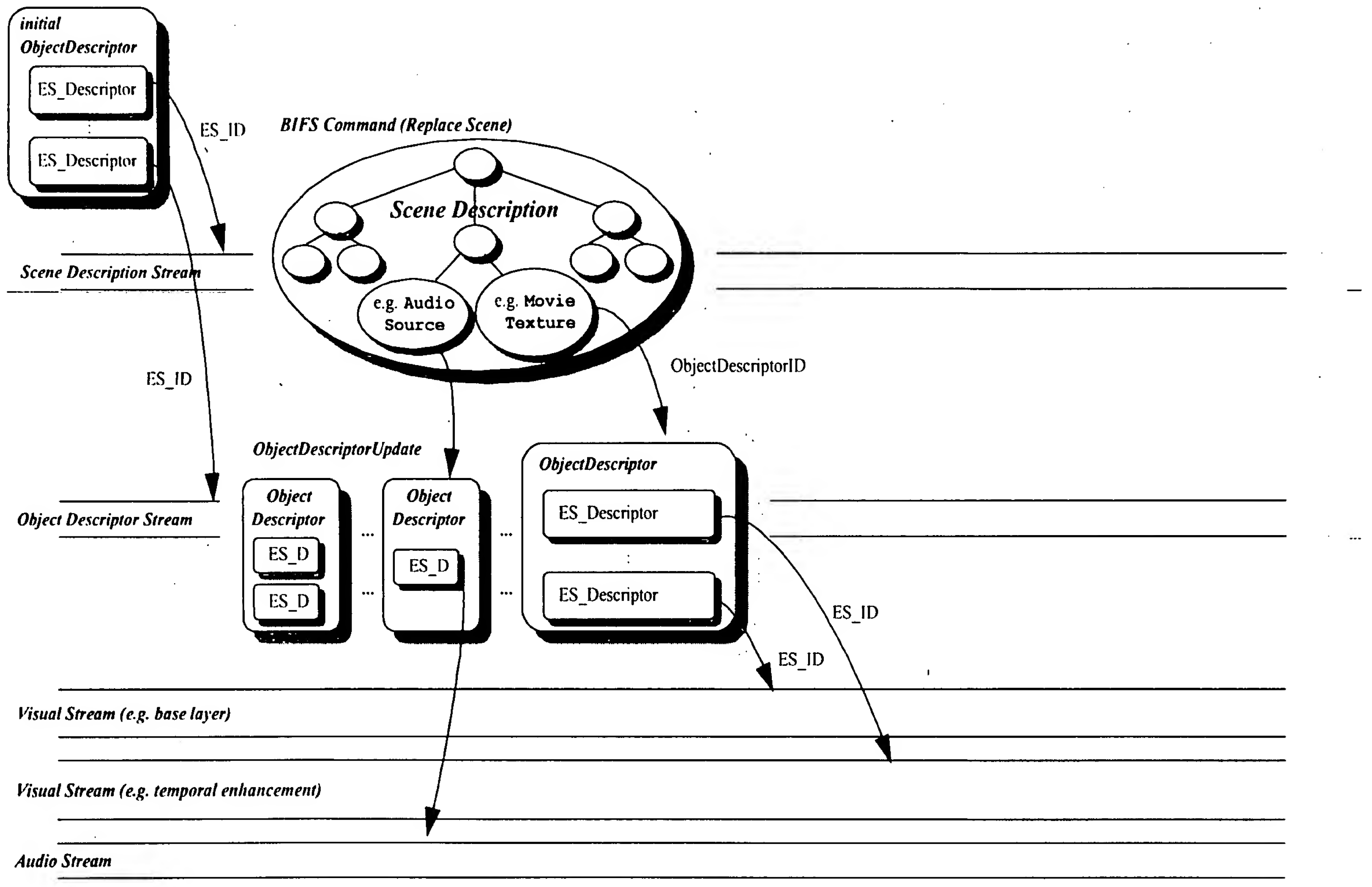


FIG. 3

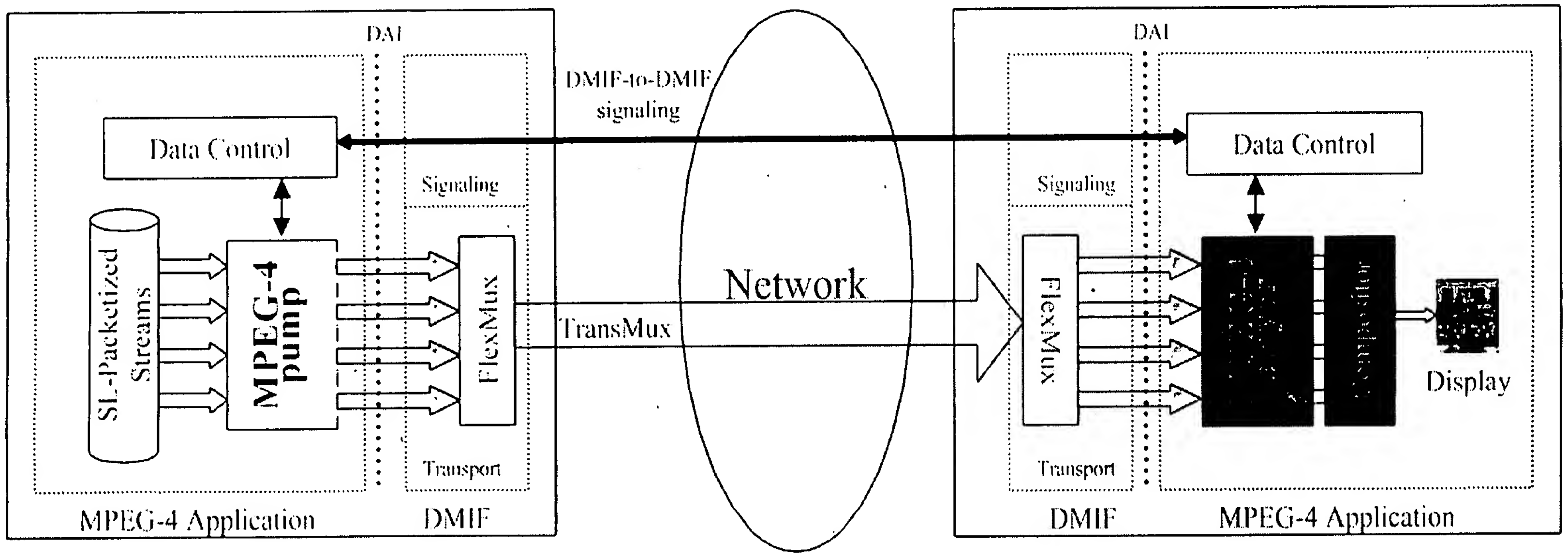
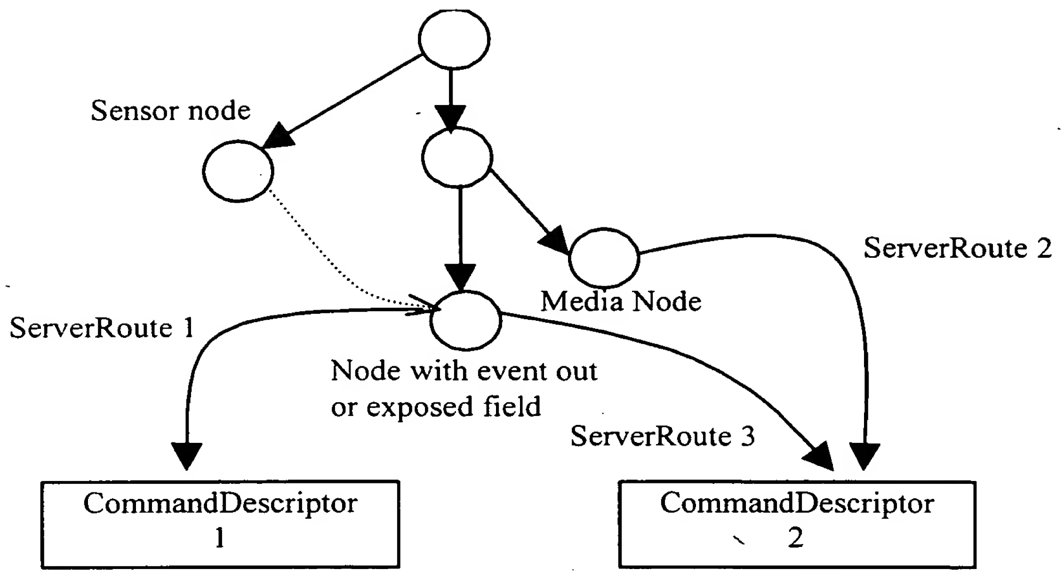
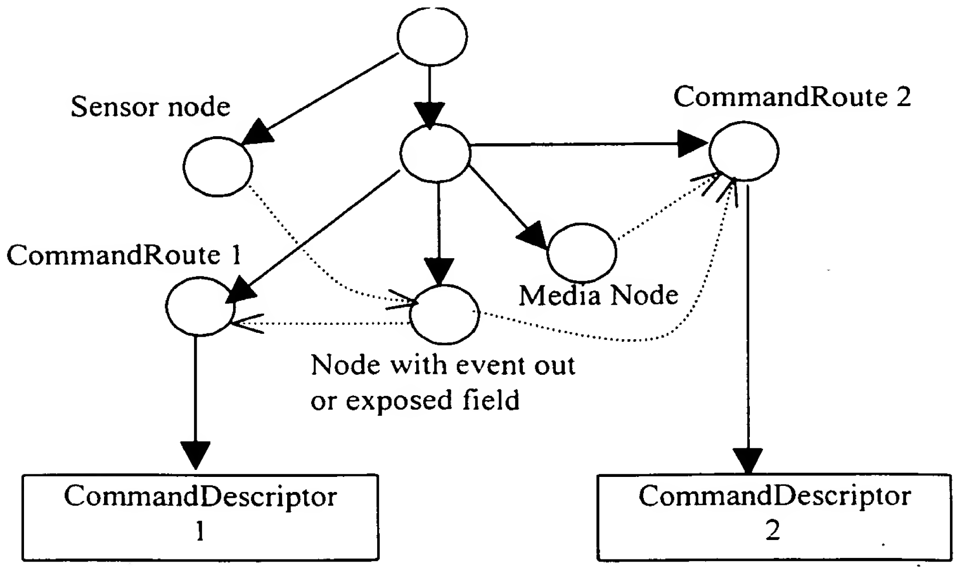

DAI
DMIF-to-DMIF signaling
Data Control
Signaling
SL-Packetized Streams
MPEG-4 pump
FlexMux
Transport
MPEG-4 Application
DMIF
Network
TransMux
DAI
Signaling
Data Control
FlexMux
Compositor
Display
Transport
DMIF
MPEG-4 Application

FIG. 4



Sensor node
ServerRoute 2
Media Node
ServerRoute 1
Node with event out
or exposed field
ServerRoute 3
CommandDescriptor
1
CommandDescriptor
2

(a)



Sensor node
CommandRoute 2
CommandRoute 1
Media Node
Node with event out
or exposed field
CommandDescriptor
1
CommandDescriptor
2

(b)

FIG. 5

```
class CommandDescriptor: bit(8) CommandDescriptorTag = 0x05 {

      bit(16) CommandDescriptorID;
      bit(16) CommandID;

      bit(16) length;

      // stream count; number of ES_IDs associated with this message
      unsigned int (8) count;

      // ES_Id (channel numbers) of the streams affected by the command
      unsigned int (16) ES_ID[count];

      // application-defined parameters
      do {
            unsigned int (8) paramLength;
            char (8) commandParam [paramLength];
      }
      while (paramLength!=0);
}
```

FIG. 6

```
class CommandDescriptorRemove: bit(8) CommandDescriptorRemoveTag = 0x06 {
      bit(16) CommandDescriptorID;
}
```

FIG. 7

```
class BIFSScene {
      SFNode nodes(SFTopNode);
      bit(1) hasROUTEs;
      if (hasROUTEs)
            ROUTEs routes ;
      bit(1) hasServerROUTEs;
      // the following is added to the MPEG-4 syntax
      if (hasServerROUTEs)
            ServerROUTEs sroutes;
}

// modification of MPEG-4 ROUTEs structure to point to command descriptor
class ServerROUTEs {

      bit(1) isUpdateable;
      if (isUpdateable)
            bit(10) srouteID;

      bit(10) outNodeID;
      NodeData nodeOUT = GetNodeFromID(outNodeID); // get source node
      int(nodeOUT.nOUTbits) outFieldRef; // event source field index

      bit(10) CD_ID; // event sink - command descriptor ID
}
```

FIG. 8

CommandRoute

Node interface

```
CommandRoute {
    eventIn        SFBool        Execute                    FALSE
    field          SFUrl         CommandDescriptor          []
}
```

Functionality and semantics

The **CommandRoute** node is used to support server interaction. A command route is executed when an event is received on the **execute** field, for example, from a touch sensor. The execution of a command route involves communicating the command pointed by the **commandDescriptor** to the server. The **commandDescriptor** field contains either a URL to the command descriptor or the ID of the Command Descriptor to be associated with this **CommandRoute** node. Commands are typically sent to a server using DMIF's DAI_User_Command primitives. The node update mechanism can be used to change the command descriptor ID. This allows supporting different interaction behavior for the same user interaction at different times (before and after node update).

FIG. 9

Command Name	Command ID
Unused	0X0000
Start	0X0001
Stop	0X0002
Pause	0X0003
Forward	0X0004
Reverse	0X0005
ContentSelection	0X0006
MPEG-4 reserved	0X0007 - 0X00FF
User Defined	0X0100 - 0XFFFF

FIG. 10

```
class Commmand {

        // command ID
        bit(16) CommandID;

        // the following are simply copied from the command descriptor
        // stream count; number of ES_IDs associated to this message
        unsigned int (8) count;

        // ES_Id of the streams
        unsigned int (16) ES_ID[count];

        // application-defined parameters
        do {
                unsigned int (8) paramLength;
                char (8) commandParam [paramLength];
        }
        while (paramLength!=0);
}
```

FIG. 11

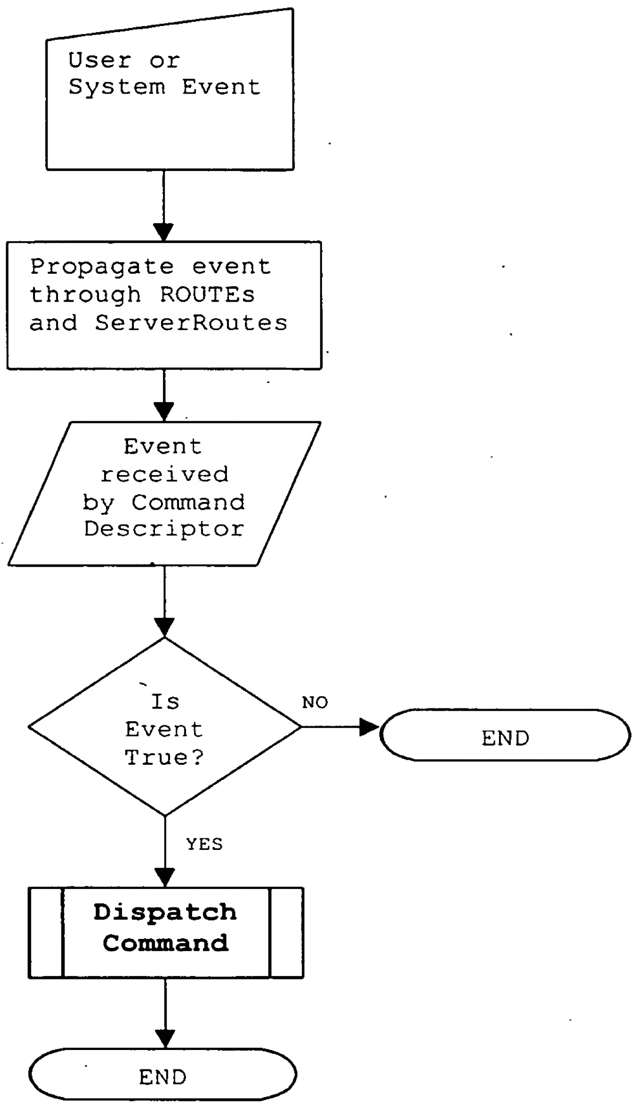

User or System Event
Propagate event through ROUTEs and ServerRoutes
Event received by Command Descriptor
Is Event True?
NO
END
YES
Dispatch Command
END

FIG. 12

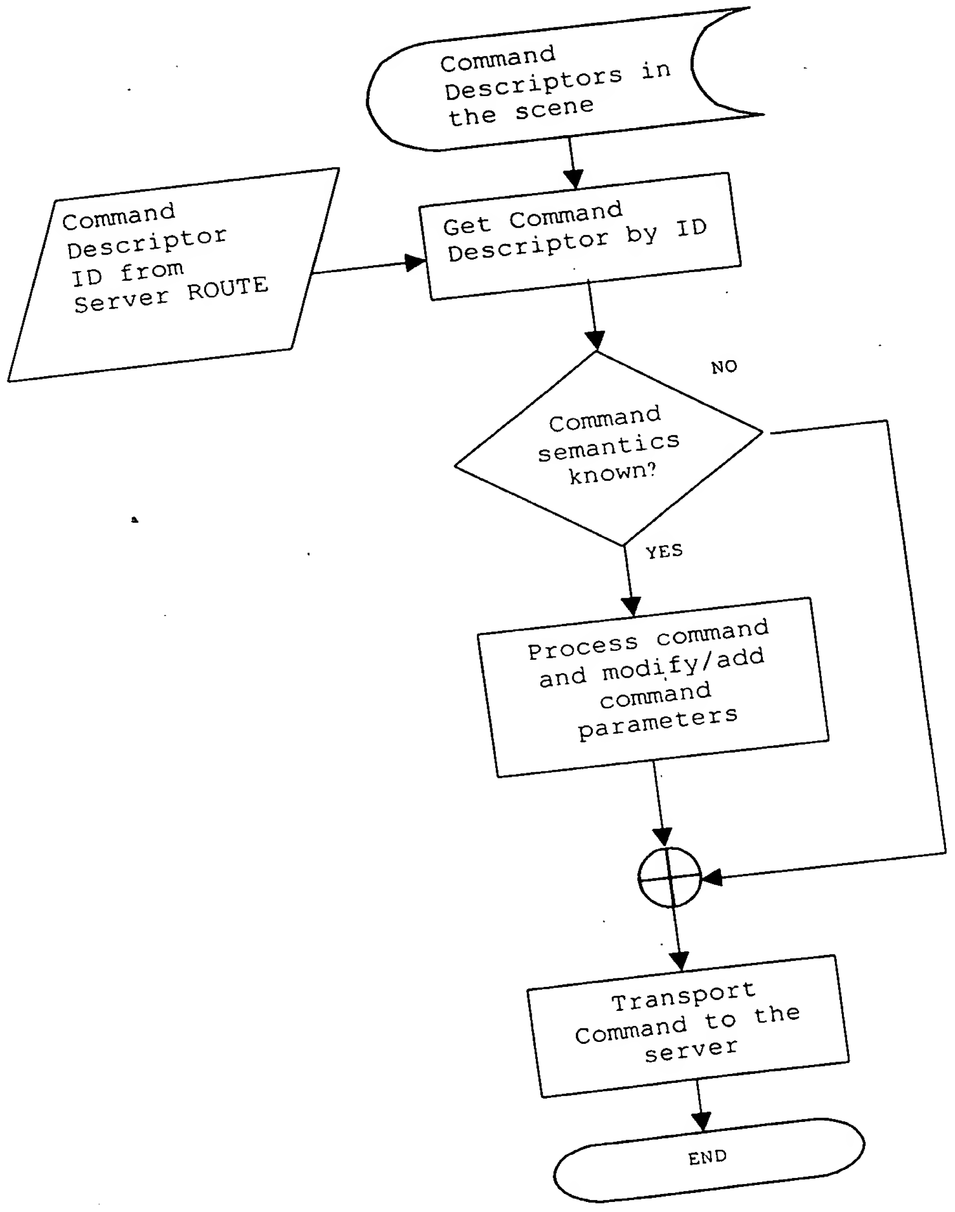

Command Descriptors in the scene
Command Descriptor ID from Server ROUTE
Get Command Descriptor by ID
Command semantics known?
NO
YES
Process command and modify/add command parameters
Transport Command to the server
END

FIG. 13

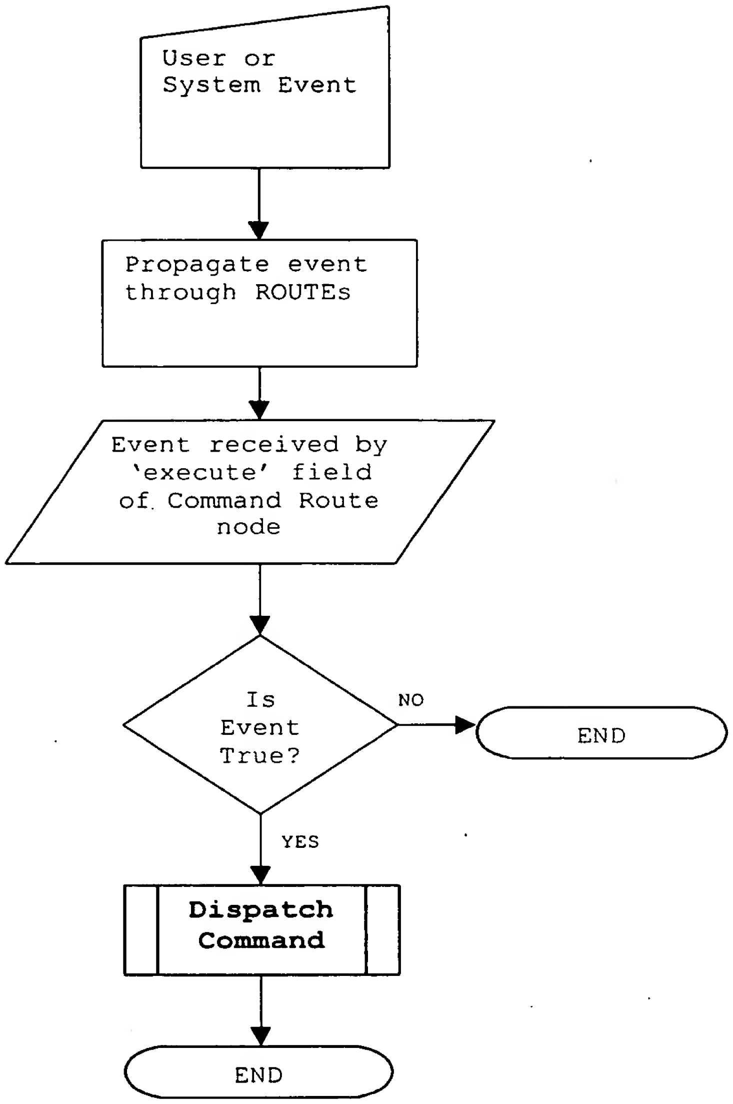

User or System Event
Propagate event through ROUTEs
Event received by 'execute' field of Command Route node
Is Event True?
NO
END
YES
Dispatch Command
END

FIG. 14

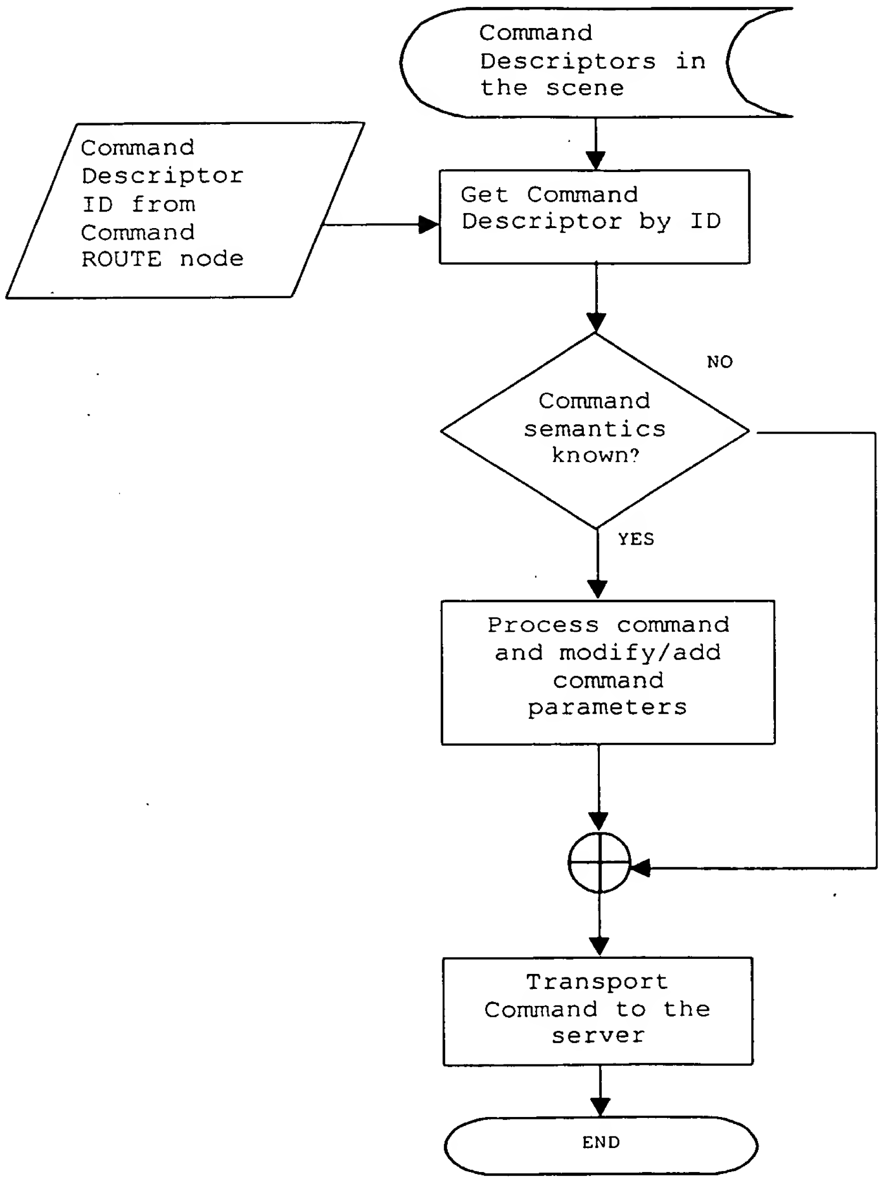

Command Descriptors in the scene
Command Descriptor ID from Command ROUTE node
Get Command Descriptor by ID
Command semantics known?
NO
YES
Process command and modify/add command parameters
Transport Command to the server
END

FIG. 15